ALAN TURING'S LEGACY

John Graham-Cumming

NewScientist

INSTANT EXPERT

23

CLASSIC STOCK/ALAMY

CARLOS BARRIA/REUTERS

THE FIRST COMPUTER

Up until the second world war, the word "computer" meant a person, often a woman, who did calculations either manually or with the help of a mechanical adding machine. These human computers were an essential part of the industrial revolution and performed often repetitive calculations, such as those necessary for the creation of books of log tables.

But in 1936, Turing, aged just 24, laid the foundations for a new type of computer - one we would still recognise today - and so played a seminal role in the information technology revolution.

Turing did not set out to invent the model for the modern computer, though. He wanted to resolve a conundrum in mathematical logic. In the mid-1930s, he decided to attack the fearsomely named Entscheidungsproblem - or "decision problem" - posed by mathematician David Hilbert in 1928.

At the time, mathematics was searching for concrete foundations and Hilbert wanted to know if all mathematical statements (such as 2 + 2 = 4) were "decidable". In other words: does a step-by-step procedure exist that can determine whether any given statement in mathematics is true or false?

This was a fundamental question for mathematicians. Although it is easy to say with certainty that a statement like 2 + 2 = 4 is true, more complex logical statements are trickier to ascertain. Take the Riemann hypothesis, proposed by Bernhard Riemann in 1859, which makes specific predictions about the distribution of prime numbers among natural numbers. Mathematicians suspect it is true, but they still don't know for certain.

If Hilbert's proposed step-by-step procedure could be found, it would mean that, eventually, a machine could be devised to give mathematicians a firm answer to any logical statement they wanted to test. All the big open questions in mathematics could be resolved.

It may not have been apparent then, but what Hilbert was searching for was a computer program. Today we call his proposed step-by-step procedure an "algorithm". But neither computers nor programs existed in the 1930s - Turing had to define the concept of computation itself in order to tackle the Entscheidungsproblem.

In 1936, Turing published a paper that provided a definitive answer to Hilbert's question: no procedure exists for determining whether any given mathematical statement is true or false. Moreover, many of the important unresolved questions in mathematics are "undecidable" (*Proceedings of the London Mathematical Society*, Vol 42, p 230). This was good news for human mathematicians, because it meant that they would never be replaced by machines. But with his paper, Turing had achieved more than the resolution of Hilbert's question.

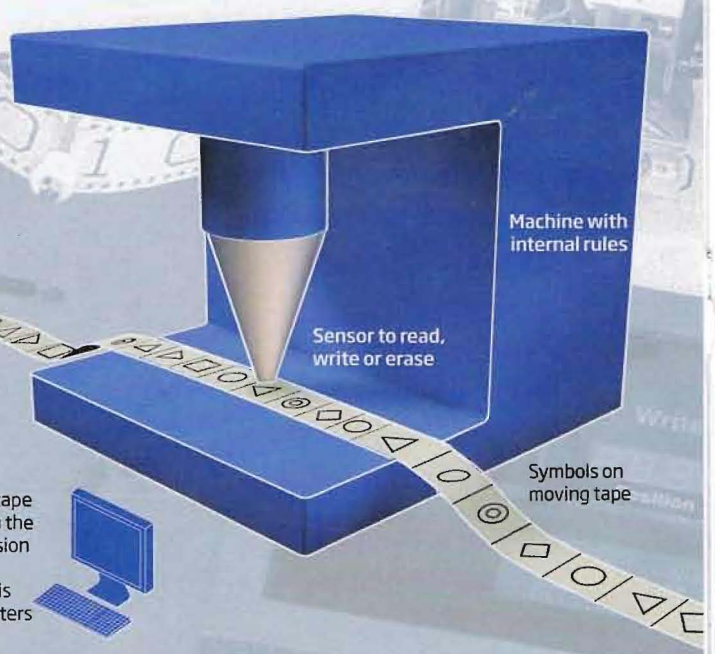
To arrive at his result, he had also come up with the theoretical basis for modern computers.

Before Turing could test Hilbert's proposal, he needed to define what a step-by-step procedure was, and the sort of device that might perform it. He did not need to build such a machine, but he did need to lay out how it would work hypothetically.

First, he imagined a machine capable of reading symbols from a paper tape. You would feed the paper tape in, and the machine would examine the symbols, then make a decision about what to do next by following a set of internal rules. It could, for example, add two numbers

Computers from the 1940s to the present day are based on Turing's model

In the 1930s, Alan Turing imagined a new type of machine that would read symbols on a tape, one at a time. After making a decision following its internal rules, it would then perform one of five actions: move the tape left or right, erase the symbol, write a new one, or stop. This is known as a Turing machine



Turing also proposed that the tape itself could be used to program the machine's actions - a basic version of software. This is called a universal Turing machine, and is the basis for all modern computers

MIKE DAVEY/ATURINGMACHINE.COM

The ideas of British scientist Alan Turing shaped our world. He laid the foundations for modern computers and the information technology revolution, as well as making far-sighted predictions on artificial intelligence, the brain and even developmental biology. He also led vital codebreaking efforts for the Allies in the second world war. This legacy will be celebrated worldwide on 23 June – the 100th anniversary of his birth.

Understanding why Turing's achievements matter today begins with the story of how he set out to solve one of his era's biggest mathematical conundrums – and in the process defined the basis of all computers

that were written on the tape and print the result further along the tape. This would later come to be known as a Turing machine (see diagram, below left). However, because each individual Turing machine had predefined internal rules – essentially a fixed program – it could not be used to test Hilbert's question.

Turing realised that it would be possible to make a machine that could initially read a procedure from the tape, and use that to define its internal rules. By doing so, it was programmable, and could perform the same actions of any individual Turing machine, which had fixed internal rules. That flexible device, which we call a universal Turing machine, is a computer.

How so? The procedure written on the tape can be thought of as software. Turing's universal machine would essentially be loading the software from the tape into itself, just as we do today with a program from a disc or download: one minute your computer is a word processor, the next it is a music player.

Once Turing had this theoretical computer, he could answer the question of what was "computable". What could a computer do and not do?

To disprove Hilbert's proposed procedure, Turing needed to find just one logical statement that a computer cannot ascertain is true or false. To do this, he identified a specific question: could a computer examine a program and decide whether it will "stop" or run forever if left unchecked? In other words, could a computer determine whether it was true or false that a program would stop? The answer, he demonstrated, is that it cannot. Hilbert's procedure therefore did not exist, and the Entscheidungsproblem was resolved. In fact, Turing's conclusion was that there are infinitely many things a computer cannot do.

"As computers get ever more advanced, they still operate within the limits Turing described"

While Turing was attacking the decision problem, US mathematician Alonzo Church was taking a pure-mathematics approach to it. Church and Turing published their papers almost simultaneously. Turing's paper defined the notion of "computable", whereas Church's had "effective calculability". The two are equivalent. This result, the Church-Turing thesis, underlies our concept of the limits of computers and creates a direct link between an esoteric question in mathematical logic and the computer you own.

As computers get ever more advanced, they operate within the same limits that Church and Turing described. Even though modern computers are stunningly powerful compared with the behemoths of the 1940s, they can still only perform the same tasks as a universal Turing machine.

PICTORIAL PRESS/ALAMY

MY, MY, MY, DELILAH

Turing is often associated with breaking codes, but in 1943 he also spent time making them. At the secret UK government laboratory at Hanslope Park in Buckinghamshire, Turing led the development of a portable device for speaking securely with another person. It was named Delilah and could be used to scramble a telephone or radio conversation.

Delilah was revolutionary because the scrambling system was very hard to break, yet was portable. By contrast, the secure phone system that linked the British prime minister at 10 Downing Street to the US president in the White House was so large that it had to be installed in the basement of the Selfridges store on London's Oxford Street. Called SIGSALY, it weighed 50 tonnes and required thousands of watts of power. Other, smaller voice scramblers were insecure and easily decoded by the Nazis.

Delilah worked by combining the speech to be scrambled with what sounded like random noise, similar to an untuned radio. When put together, all an eavesdropper would hear was noise, but by careful synchronisation between two Delilah machines at either end, it was possible to filter out the random noise and hear the original speech.

The system relied on two techniques that are common in modern codes: the creation of an apparently random stream of numbers, and modular arithmetic.

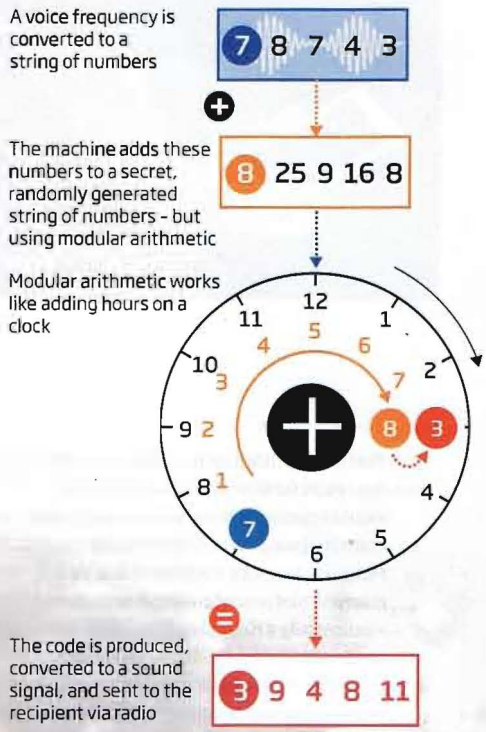
Delilah generated a sequence of numbers that both the sender and receiver could reproduce from a secret key (a way of setting the machine known only to them). By sampling the waveform of the voice to be transmitted - just as modern computers sample music to produce numbers that are stored on a CD or in an MP3 music file - the voice became a stream of numbers. These could then be enciphered by adding each number to a corresponding number from the random sequence. At the other end, a subtraction of the same random number would reveal the original number, which could then be used to reproduce the voice waveform.

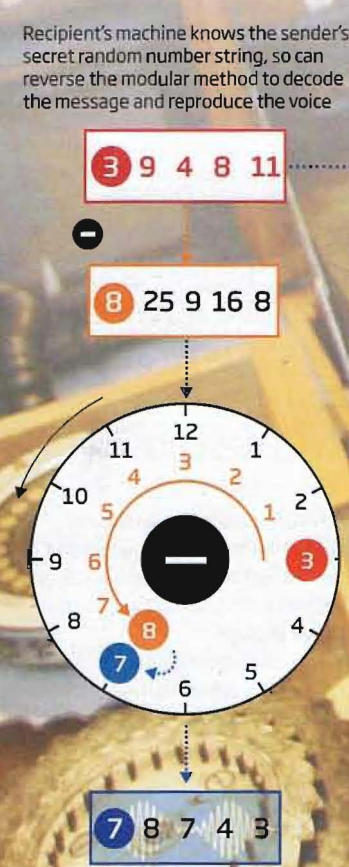
What made this secure was that the addition and subtraction was not applied using standard linear arithmetic, but instead a special form called "modular arithmetic".

Modular arithmetic works a bit like a clock, on which at some point the numbers wrap around (see diagram, right). On a clock, once you pass 12 (or 24) the numbers begin again. When the numbers used to encode information wrap around in this way, it becomes hard to determine what numbers were used in any calculation just by looking at the answer.

For example, consider how the time 2200 is 3 hours before 0100, but it is also 27 hours before 0100, and 51 hours before 0100 and so on. This wrap-around means that there is an infinity of numbers for any addition or subtraction making it very hard for a codebreaker to determine which one was used.

Alan Turing invented a portable machine called Delilah that encrypted voice messages using a mathematical approach called "modular arithmetic"





"Enigma messages were impossible to crack using raw manpower or simple mechanical calculators"

NESTOR BACHMANN/DPA/PRESS ASSOCIATION IMAGES

During the second world war, Alan Turing was recruited by the Allies to crack the coded messages used by the Nazis. At first they seemed near-impossible to decipher, but using a combination of mathematics and engineering, he and his colleagues found a way, and so shaped the course of the war.

During this period, Turing also devised a portable machine called Delilah that could securely encode a voice message. It was based on a special form of arithmetic and was years ahead of its time

CRACKING NAZI CODES

When studying at Princeton University in 1936, Turing wrote to his mother saying that he had discovered a way to use mathematics to encrypt messages. Three years later, he was back in the UK using his skills to break Nazi codes. His efforts at the British military intelligence base Bletchley Park significantly affected the course of the second world war.

At Bletchley, Turing was known universally as "The Prof". He quickly became arguably the most important code breaker there. Most famously, he helped decipher the messages created by the Nazi's Enigma coding machines. To do so, he and his colleagues had to use a combination of insight, intelligent guesses and clever engineering.

An Enigma machine employed rotating wheels that assigned each typed letter to a different coded letter of the alphabet. A second machine at the receiver's end would reverse the process, deciphering the message.

Enigma messages were impossible to crack by brute force, using either raw manpower or simple mechanical calculators, because of the huge number of "keys" used. A key is the crux of any code system: it is a secret "password" agreed on by two people communicating in code. In the case of Enigma, the key consisted of the way each machine was set up before a message was transmitted. Both sender and receiver would arrange their wheels in a pre-agreed fashion, as well as arranging a plugboard similar to that used in a telephone exchange. Depending on whether three or four wheels were used, that meant there were 26x26x26 (17,576) or 26x26x26x26 (456,976) possible keys. And the additional variations in the plugboard settings made messages even more secure.

For the Allies, breaking Enigma was essential to winning the battle of the Atlantic and stemming the huge loss of ships and people inflicted by Nazi submarines. It was a particularly difficult task, because the German navy's U-boats used the four-wheel Enigma, and changed the key frequently. Working with Gordon Welchman, Turing devised a scheme that could reduce the number of possible keys that needed to be tried. It relied on cleverly chosen "cribs" - essentially guesses at part of the contents of a message. A codebreaker might, for example, guess that the message contained the name of a city or a particular ship. That name would be a crib.

Turing and Welchman designed a machine called the Bombe, which was capable of running through the possible keys of an Enigma machine, checking for a match between a crib and a portion of the encrypted message. In other words, if the name of a ship was indeed part of a secret message, the Bombe was capable of spotting it. The Bombe was able to quickly identify which keys were likely candidates because it incorporated shortcuts based on the mathematics of Enigma machines that had been captured. The Bombe's wheel and plugboard settings could then be used to decipher the rest of message.

Turing's work at Bletchley reached far beyond Enigma and the Bombe. The Nazis were using a high-speed encrypted teleprinter system that the British called Tunny for the most secret messages. Mathematician Bill Tutte determined, in a brilliant insight, how the machine worked without ever having seen one. Turing then developed a scheme to attack the Tunny cipher, called Turingismus, and by 1943, high-level messages between Hitler and his top commanders were being read. Others, including engineer Tommy Flowers, went on to build Colossus, an early computer that operated at lightning speeds to decipher Tunny.

In 1943, Turing moved over to the secret base at Hanslope Park, Buckinghamshire, to work on a system for secure speech communications called Delilah. Only recently have the full details of this endeavour been declassified (see "My, my, my, Delilah", above left.)

MARK FARRINGTON

Reconstructed: the Bombe machine's dials, used to break Nazi Enigma codes

ARTIFICIAL BRAINS

Turing was curious about the brain. He believed that the infant brain could be simulated on a computer. In 1948, he wrote a report arguing for his theory, and in doing so gave an early description of the artificial neural networks used to simulate neurons today.

His paper was prescient, but was not published until 1968 - years after his death - in part because his supervisor at the National Physical Laboratory, Charles Galton Darwin, described it as a "schoolboy essay".

The paper describes a model of the brain based on simple processing units - neurons - that take two inputs and have a single output. They are connected together in a random fashion to make a vast network of interconnected units. The signals, passing along interconnections equivalent to the brain's synapses, consisted of 1s or 0s. Today this is called a "boolean neural network"; Turing called it an unorganised A-type machine.

The A-type machine could not learn anything, so Turing used it as the basis for a teachable B-type machine. The B-type was identical to the A-type except that the interconnections between neurons had switches that could be "educated". The education took the form of telling a switch to be on (allowing a signal to pass down the synapse) or off (blocking the signal). Turing theorised that such education could be used to teach the network of neurons.

After his death, Turing's ideas were rediscovered and his simple binary-based neural networks were shown to be teachable. For example, they can learn to recognise simple patterns like the shapes of Os and Xs.

Later, independently, more complex neural networks became the focus of AI research and they are used to this day in robotics, pattern recognition and game playing.

E.M. PASIEKA/CORBIS

"Turing counted the number of petals on daisies and found they always added up to a Fibonacci number"

MATHEMATICAL BIOLOGY

DANITA DELIMONT/GETTY

Same cow, different pattern

How does the leopard get its spots? Why is the black-and-white pattern on the skin of a Friesian cow haphazard and asymmetrical? These were questions that intrigued Turing. In the 1950s, he found an answer that had previously eluded biologists.

From childhood, Turing had been fascinated by mathematical patterns in nature, and, in particular, the recurrence of the Fibonacci sequence in plants. The Fibonacci sequence is 1, 1, 2, 3, 5, 8, 13 and so on. Each number in the sequence is the sum of the previous two. Turing had counted the number of petals on daisies, and found the total would always add up to a Fibonacci number. The same applies to other flowers. He believed that the spiral pattern on the head of a sunflower was determined by the Fibonacci sequence.

As an adult, Turing became curious about a biological concept called morphogenesis - essentially, how does a living body take shape as it grows? At the time, a fundamental practical question for life was: how did a single, simple cell with a simple shape take on the complex form of a living being just by dividing? Why didn't division simply result in ever more copies of the same cell? How did odd shapes like hands or eyes develop?

In attempting to answer those questions, Turing turned to mathematics. He posited the existence of chemicals he termed "morphogens" that cause the asymmetry seen in living beings. His 1952 paper "The chemical basis of morphogenesis" is now seen as the starting point for an entire branch of biology (*Philosophical Transactions of the Royal Society B*, vol 237, p 37).

Turing believed that specific chemical reactions were responsible for the irregular spots and patches on the skin of animals like leopards or cows, and the ridges inside the roof of the mouth. So he used the first computers to simulate the process that he thought might be occurring.

Turing modelled a pair of interacting chemicals undergoing diffusion and reaction. The two chemicals diffuse, or spread out,

Alan Turing was a visionary thinker on artificial intelligence (AI), devising the Turing test, which is still used as a key gauge of how close machines have come to human intelligence. He also published prescient ideas about simulating a brain with computers.

Toward the end of his life, he was also beginning tantalising work in biology, devising a mathematical theory of "morphogenesis" – in essence, how a leopard gets its spots

at different rates resulting in varying concentrations of the two. They also react with one another to produce more of the same chemicals, which in turn diffuse and react.

Turing's simple combination of reaction and diffusion results in striking, irregular patterns. He theorised that if one of the chemicals caused activity in cells, and another prevented it, the result would be the types of patterns seen on the coats of animals. In his 1952 paper, he revealed a computer simulation of reaction and diffusion that resulted in patterns similar to those seen on Friesian cows.

But his idea remained just a theory. Actual morphogens were not found during his lifetime. In January 2012, a group of researchers at King's College London showed that Turing had been right and that such reactions do exist in nature. They showed that two chemicals, acting as Turing predicted, control the formation of ridge patterns inside a mouse's mouth (*Nature Genetics*, vol 44, p 348).

THE TURING TEST

In 1950, Turing described what we now call the Turing test. To this day, his test is a standard by which "intelligent" machines are judged, and it is remarkable in its simplicity and ingenuity.

Turing referred to his method of determining whether a machine could be called intelligent as the Imitation Game. He proposed that a machine would be intelligent if it could not be distinguished from a human (*Mind*, vol 59, p 433).

In his test, a judge communicates with both a human and a machine in written language, via a computer screen or teleprinter. This means the judge can use only the conversation to determine which is which. If the judge cannot distinguish the machine and the human, the machine is deemed to be intelligent.

The concept will be familiar to anyone who has interacted with an artificial intelligence such as Apple's digital personal assistant, Siri, or a chatbot. Siri does not pass the test, and although chatbots may have fooled some individuals in recent years, none have passed the Turing test unequivocally. In fact, the limitations of even the best modern AIs mean that they are quickly outed as machines. Still, Turing imagined a day when AI would prove indistinguishable from the human form.

OLIS/AFP/GETTY

Siri on the iPhone is smart, but not smart enough to pass for a human

JEREMY BECHTEL

John Graham-Cumming
John Graham-Cumming is a programmer, amateur codebreaker and writer based in London. He is the author of *The Geek Atlas* (O'Reilly Media, 2009). In 2009, he successfully campaigned for an official apology for Alan Turing from the UK government

NEXT
INSTANT
EXPERT

Steve Haake
SPORTS ENGINEERING
7 July

LIFE, INTERRUPTED

Alan Turing was undoubtedly one of the greatest intellects of the 20th century. In January, *Nature* called him "one of the top scientific minds of all time". It is easy to agree with that evaluation.

Turing essentially founded computer science, helped the Allies win the second world war with hard work and a succession of insights, asked fundamental questions about the nature of intelligence and its link with the brain's structure, and laid the foundations for an area of biology that is only now being fully appreciated and researched.

But this wide-ranging, original and deep mind was lost in 1954 when he took his own life following his conviction for "gross indecency" – essentially, for being a practising homosexual, which was illegal in the UK at that time.

Turing died when computers were in their bulky infancy, when the structure of DNA had just been unravelled by Francis Crick and James Watson, and before artificial intelligence even had a name.

Turing's record languished in relative obscurity until the 1970s – partly because of his homosexuality and suicide, and partly because of the deeply mathematical papers he produced and the secrecy surrounding his work at Bletchley Park.

After homosexuality was decriminalised in the UK in 1967, and the secrets of Bletchley Park were revealed, Turing's legacy began to be recognised. Then in 1983, mathematician Andrew Hodges at the University of Oxford published the definitive biography of Turing (see "Further reading", right) and Turing and his achievements became widely known and appreciated beyond academia.

Looking back now at the 41 years of Turing's life and his continuing impact, we can only wonder what he would have turned his singular mind to next, had he lived the long and rich life he deserved.

FURTHER READING

Alan Turing: The enigma by Andrew Hodges, Vintage, Random House, 1983

The Annotated Turing by Charles Petzold, John Wiley & Sons, 2008

The Essential Turing by B. Jack Copeland, Clarendon Press, 2004

Alan M. Turing: Centenary edition by Sara Turing, Cambridge University Press, 2012

The Code Book: The secret history of codes and code-breaking by Simon Singh, Fourth Estate, 2002

WEBSITES

Alan Turing centenary year events and information: turingcentenary.eu

Bletchley Park: www.bletchleypark.org.uk

Turing's reaction-diffusion model of morphogenesis: cgjennings.ca/toybox/turingmorph

Cover image: Mark Farrington